README

Andrea Pierré

December 16, 2018

Contents

1 Design decisions

1.1 Data

For consistency reasons since *Node.js* was an imposed choice for the back-end, and to not impose to my reviewer to install another language, I would have chosen to load the data in *Javascript*. But since I was allowed to use Docker this was not a problem anymore. So since time was limited, I choose Python to get the data in the database as it was the language I was more confident with.

The *MySQL* official Docker image was more than 100MB, I thought it was overkill for a simple application like this one, so I eliminated *MySQL*. I surprisingly found a lean *alpine* version of *PostgreSQL* which was less than 30MB, so I hesitated between *PostgreSQL* and *SQLite*. At the end I chose to go with *PostgreSQL* because it was simpler to use with Docker. Without Docker I would have chosen to go with *SQLite*. I also chose to go with the *SQLAlchemy* ORM in case I had some problem down the road so that it

would be easy to switch to another database in case (and also because I wanted to learn it).

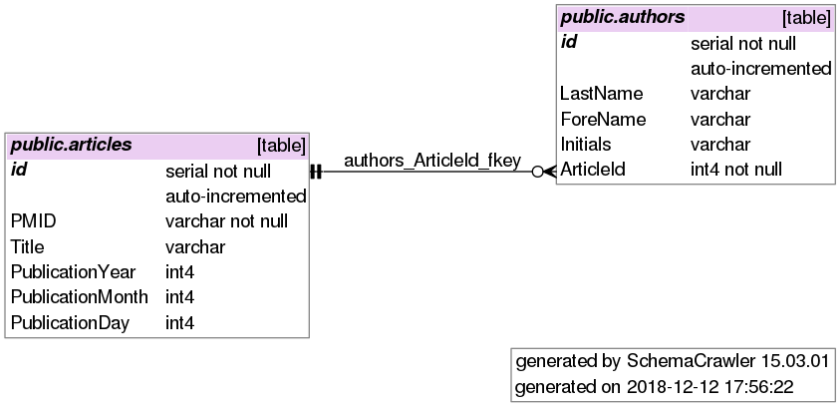

Figure 1: Entity relationship diagram of the database

When the app is running, you can inspect the database with the *Adminer* web client running in a container using the following URL: `http://localhost:8081/?pgsql=db&username=postgres&db=postgres&ns=public`

1.2 Back-end

The assignment asked for a single page application, the back-end only being there to make the queries to the database, so I believe a simple *Node.js* API with the following three routes should do the work:

- one route to get the list of the articles,
- one route to search by author last name,
- one route to get the data for the visualization.

I used the *Sequelize* ORM to query the database.

1.3 Front-end

Since I have a really small experience with front-end frameworks, and since time was limited, I choose the one I read it had the more gentle learning curve, e.g. *Vue.js*. Without the time limiting constraint, I would probably have chosen *React* which has the biggest community today. I also used the

Vuetify plugin to gain some time with already tuned components with nice CSS.

For the visualization, I chose to use the *Britecharts* library, which is built on top of *D3.js*, and which should require less time to learn how to use. Without the time limiting constraint, I would probably have chosen to go directly with *D3.js*.

2 Setup the app

Just run `docker-compose up` and open your browser at the following URL: `http://localhost:8080/`

3 Time spent

Table 1: Time spent on assignment

Design decisions	2h
Pulling data from PubMed API	2h
Database design and data parsing	6h
Back-end development	3h
Learning front-end framework	2h
Front-end development	
Docker containerization	4h

4 FDA 21 CFR 820.30